What Is Claimed Is:

1. A method for dynamically adjusting the aggressiveness of an execute-ahead processor, comprising:

executing instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred, and other non-deferred instructions are executed in program order, and wherein if a non-data-dependent stall condition is encountered, the execute-ahead processor enters a scout mode, wherein instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the execute-ahead processor;

if an unresolved data dependency is resolved during the execute-ahead mode, executing deferred instructions in a deferred mode;

wherein if some instructions are deferred again during the deferred mode, the method further comprises,

determining whether to resume execution in the execute-ahead mode,

if it is determined to do so, resuming execution in the execute-ahead mode, and

otherwise resuming execution in a non-aggressive mode.

2. The method of claim 1, wherein resuming execution in the non-aggressive execution mode involves remaining in the deferred mode until all deferred instructions are executed and the execute-ahead processor returns to a normal execution mode.

3. The method of claim 1, wherein resuming execution in the non-aggressive mode involves resuming execution in a non-aggressive execute-ahead mode, wherein if a non-data-dependent stall condition is encountered, the execute-ahead processor does not enter the scout mode, but instead waits for the non-data-dependent stall condition to be resolved, or for an unresolved data dependency to return, before proceeding.

4. The method of claim 1, wherein prior to executing instructions in execute-ahead mode, the method further comprises entering the execute-ahead mode by:

issuing instructions for execution in program order during a normal execution mode;

upon encountering an unresolved data dependency during execution of an instruction,

generating a checkpoint that can subsequently be used to return execution to the point of the instruction, and

executing subsequent instructions in the execute-ahead mode.

5. The method of claim 4, wherein if the launch point stall condition (the unresolved data dependency or the non-data-dependent stall condition that originally caused the execute-ahead processor to exit the normal execution mode) is finally resolved, the method further comprises using the checkpoint to resume execution in the normal execution mode from the launch point instruction (the instruction that originally encountered the launch point stall condition).

6. The method of claim 1, wherein executing deferred instructions in the deferred mode involves:

issuing deferred instructions for execution in program order;

deferring execution of deferred instructions that still cannot be executed because of unresolved data dependencies; and

executing other deferred instructions that are able to be executed in program order.

7. The method of claim 6, wherein if all deferred instructions are executed in the deferred mode, the method further comprises returning to a normal execution mode to resume normal program execution from the point where the execute-ahead mode left off.

8. The method of claim 1, wherein the unresolved data dependency can include:

a use of an operand that has not returned from a preceding load miss;

a use of an operand that has not returned from a preceding translation lookaside buffer (TLB) miss;

a use of an operand that has not returned from a preceding full or partial read-after-write (RAW) from store buffer operation; and

a use of an operand that depends on another operand that is subject to an unresolved data dependency.

9. The method of claim 1, wherein the non-data-dependent stall condition can include:

a memory barrier operation;

a load buffer full condition; and

a store buffer full condition.

10. An apparatus that dynamically adjusts the aggressiveness of an execute-ahead processor, comprising:

an execution mechanism configured to execute instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred, and other non-deferred instructions are executed in program order, and wherein if a non-data-dependent stall condition is encountered, the execution mechanism is configured to enter a scout mode, wherein instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the execute-ahead processor;

wherein if an unresolved data dependency is resolved during the execute-ahead mode, the execution mechanism is configured to execute deferred instructions in a deferred mode;

wherein if some instructions are deferred again during the deferred mode, the execution mechanism is configured to,

determine whether to resume execution in the execute-ahead mode,

if it is determined to do so, to resume execution in the execute-ahead mode, and

otherwise to resume execution in a non-aggressive mode.

11. The apparatus of claim 10, wherein while resuming execution in the non-aggressive execution mode, the execution mechanism is configured to remain in the deferred mode until all deferred instructions are executed and the execution mechanism returns to a normal execution mode.

12. The apparatus of claim 10, wherein while resuming execution in the non-aggressive execution mode, the execution mechanism is configured to resume execution in a non-aggressive execute-ahead mode, wherein if a non-data-dependent stall condition is encountered, the execution mechanism does not enter the scout mode, but instead waits for the non-data-dependent stall condition to be resolved, or for an unresolved data dependency to return, before proceeding.

13. The apparatus of claim 10, wherein prior to executing instructions in execute-ahead mode, the execution mechanism is configured to enter the execute-ahead mode by:

issuing instructions for execution in program order during a normal execution mode;

upon encountering an unresolved data dependency during execution of an instruction,

generating a checkpoint that can subsequently be used to return execution at to the point of the instruction, and

executing subsequent instructions in the execute-ahead mode.

14. The apparatus of claim 13, wherein if the launch point stall condition (the unresolved data dependency or the non-data-dependent stall condition that originally caused the execution mechanism to exit the normal execution mode) is finally resolved, the execution mechanism is configured to use the checkpoint to resume execution in the normal execution mode from the launch point instruction (the instruction that originally encountered the launch point stall condition).

15. The apparatus of claim 10, wherein while executing deferred instructions in the deferred mode, the execution mechanism is configured to:

issue deferred instructions for execution in program order;

defer execution of deferred instructions that still cannot be executed because of unresolved data dependencies; and to

execute other deferred instructions that are able to be executed in program order.

16. The apparatus of claim 15, wherein if all deferred instructions are executed in the deferred mode, the execution mechanism is configured to return to a normal execution mode to resume normal program execution from the point where the execute-ahead mode left off.

17. The apparatus of claim 10, wherein the unresolved data dependency can include:

a use of an operand that has not returned from a preceding load miss;

a use of an operand that has not returned from a preceding translation lookaside buffer (TLB) miss;

a use of an operand that has not returned from a preceding full or partial read-after-write (RAW) from store buffer operation; and

a use of an operand that depends on another operand that is subject to an unresolved data dependency.

18. The apparatus of claim 10, wherein the non-data-dependent stall condition can include:

a memory barrier operation;

a load buffer full condition; and

a store buffer full condition.

19. A computer system that dynamically adjusts the aggressiveness of an execute-ahead processor, comprising:

an execute-ahead processor;

a memory;

an execution mechanism within the execute-ahead processor configured to execute instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred, and other non-deferred instructions are executed in program order, and wherein if a non-data-dependent stall condition is encountered, the execution mechanism is configured to enter a scout mode, wherein instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the execute-ahead processor;

wherein if an unresolved data dependency is resolved during the execute-ahead mode, the execution mechanism is configured to execute deferred instructions in a deferred mode;

wherein if some instructions are deferred again during the deferred mode, the execution mechanism is configured to,

determine whether to resume execution in the execute-ahead mode,

if it is determined to do so, to resume execution in the execute-ahead mode, and

otherwise to resume execution in a non-aggressive mode.

20. The computer system of claim 19, wherein while resuming execution in the non-aggressive execution mode, the execution mechanism is

configured to remain in the deferred mode until all deferred instructions are executed and the execution mechanism returns to a normal execution mode.

21. The computer system of claim 19, wherein while resuming execution in the non-aggressive execution mode, the execution mechanism is configured to resume execution in a non-aggressive execute-ahead mode, wherein if a non-data-dependent stall condition is encountered, the execution mechanism does not enter the scout mode, but instead waits for the non-data-dependent stall condition to be resolved, or for an unresolved data dependency to return, before proceeding.